

**Appl. No. 10/682,041
Amdt. dated May 26, 2005
Reply to Office action of March 17, 2005**

The Pending Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Previously presented) A method for delaying asynchronous writes in a distributed file system to which a unique identifier is assigned, comprising:
buffering a page of dirty data with the unique identifier upon writing to a server;
changing the unique identifier to create a current unique identifier that is assigned to the distributed file system upon a failure of the server;
comparing the buffered unique identifier with the current unique identifier when the page is requested while the page is in a written state; and
handling the request responsive to the comparison.
2. (Original) The method of claim 1, further comprising assigning the unique identifier.
3. (Previously presented) The method of claim 2, wherein assigning the unique identifier comprises assigning a sequence number.
4. (Original) The method of claim 3, wherein assigning the sequence number comprises assigning a cluster-wide sequence number.
5. (Original) The method of claim 3, wherein changing the unique identifier comprises incrementing the sequence number.
6. (Previously presented) The method of claim 1, wherein buffering the dirty page comprises storing the unique identifier in a header of the page.

**Appl. No. 10/682,041
Amdt. dated May 26, 2005
Reply to Office action of March 17, 2005**

7. (Previously presented) The method of claim 6, wherein buffering the written page with the unique identifier comprises returning the page to a least recently used queue.
8. (Previously presented) The method of claim 1, wherein buffering the written page with the unique identifier comprises returning the page to a least recently used queue.
9. (Original) The method of claim 1, wherein the dirty page transitions to a written state upon the write to the server.
10. (Original) The method of claim 9, wherein the written page is handled as though in the dirty state when the page is in the written state.
11. (Previously presented) The method of claim 1, wherein buffering the written page with the unique identifier comprises caching the written page with the unique identifier.
12. (Previously presented) The method of claim 11, wherein caching the written page with the unique identifier comprises caching the written page with the unique identifier on the client.
13. (Previously presented) The method of claim 1, wherein changing the unique identifier comprises incrementing the unique identifier.
14. (Previously presented) The method of claim 1, wherein comparing the buffered unique identifier with the current unique identifier comprises determining whether the current unique identifier is a numerically higher value.

Appl. No. 10/682,041
Amdt. dated May 26, 2005
Reply to Office action of March 17, 2005

15. (Previously presented) The method of claim 1, wherein handling the request responsive to the comparison comprises:

storing the buffered written page to disk storage if the buffered unique identifier differs from the current unique identifier; or
performing a file sync operation if the buffered unique identifier matches the current unique identifier.

16. (Original) The method of claim 1, further comprising:

buffering a plurality of dirty pages before writing them to the server; and
writing the buffered dirty pages to the server in a single write operation.

17. (Original) The method of claim 1, further comprising:

buffering a plurality of written pages; and
writing the plurality of written pages to disk storage in a single write operation.

18. (Canceled).

19. (Previously presented) The method of claim 1, further comprising maintaining cache consistency and wherein maintaining cache consistency comprises:

issuing an exclusive mode token from the server to a client to permit the client to dirty the page;
issuing a shared mode token from the server to the client to permit the client to use, but not dirty, the page;
revoking the exclusive mode token before issuing the shared mode token.

20. (Original) The method of claim 19, wherein at least one of the exclusive mode token and the shared mode token is embedded in at least one of a read and a write operations in the operating system.

Appl. No. 10/682,041
Amdt. dated May 26, 2005
Reply to Office action of March 17, 2005

21. (Previously presented) The method of claim 19, wherein revoking the exclusive mode token comprises revoking the exclusive mode token when another computing system wants to read the data.

22. (Original) The method of claim 19, further comprising a server sharing free space information with a plurality of clients.

23. (Canceled).

24. (Previously presented) The method of claim 1, further comprising ensuring that a file modification time of a file is updated before or at file close time but not after, comprising:

- setting a flag at the client when any page of the file is dirtied;
- clearing the flag when dirty data is sent to the server before the file is closed;
- notifying the server from the client to update the modification time if the flag is still set at close time;
- notifying the server from the client to forego updating the modification time if dirty data is sent to the server after the file has been closed.

25. (Previously presented) A program storage medium encoded with instructions that, when executed by a computer, perform a method for delaying asynchronous writes in a distributed file system to which a unique identifier is assigned, the method comprising:

- buffering a page of dirty data with the unique identifier upon writing to a server;
- changing the unique identifier upon a failure of the server;
- comparing the buffered unique identifier with the current unique identifier when the page is requested while the page is in a written state; and
- handling the request responsive to the comparison.

**Appl. No. 10/682,041
Amdt. dated May 26, 2005
Reply to Office action of March 17, 2005**

26. (Original) The program storage medium of claim 25, wherein the method further comprises assigning the unique identifier.
27. (Previously presented) The program storage medium of claim 26, wherein assigning the unique identifier in the encoded method comprises assigning a sequence number.
28. (Original) The program storage medium of claim 27, wherein assigning the sequence number in the encoded method comprises assigning a cluster-wide sequence number.
29. (Original) The program storage medium of claim 27, wherein changing the unique identifier in the encoded method comprises incrementing the sequence number.
30. (Previously presented) The program storage medium of claim 25, wherein buffering the dirty page in the encoded method comprises storing the unique identifier in a header of the page.
31. (Previously presented) The program storage medium of claim 30, wherein buffering the written page with the unique identifier in the encoded method comprises returning the page to a least recently used queue.
32. (Previously presented) The program storage medium of claim 25, wherein buffering the written page with the unique identifier in the encoded method comprises returning the page to a least recently used queue.
33. (Original) The program storage medium of claim 25, wherein the dirty page transitions to a written state upon the write to the server in the encoded method.

**Appl. No. 10/682,041
Amdt. dated May 26, 2005
Reply to Office action of March 17, 2005**

34. (Original) The program storage medium of claim 33, wherein the written page is handled by the encoded method as though in the dirty state when the page is in the written state.

35. (Previously presented) The program storage medium of claim 25, wherein buffering the written page with the unique identifier in the encoded method comprises caching the written page with the unique identifier.

36. (Previously presented) The program storage medium of claim 35, wherein caching the written page with the unique identifier in the encoded method comprises caching the written page with the unique identifier on the client.

37. (Previously presented) The program storage medium of claim 25, wherein changing the unique identifier in the encoded method comprises incrementing the unique identifier.

38. (Previously presented) The program storage medium of claim 25, wherein comparing the buffered unique identifier with the current unique identifier in the encoded method comprises determining whether the current unique identifier is a numerically higher value.

39. (Previously presented) The program storage medium of claim 25, wherein handling the request responsive to the comparison in the encoded method comprises:

storing the buffered written page to disk storage if the buffered unique identifier differs from the current unique identifier; or
performing a file sync operation if the buffered unique identifier matches the current unique identifier.

**Appl. No. 10/682,041
Amdt. dated May 26, 2005
Reply to Office action of March 17, 2005**

40. (Original) The program storage medium of claim 25, wherein the encoded method further comprises:

buffering a plurality of dirty pages before writing them to the server; and
writing the buffered dirty pages to the server in a single write operation.

41. (Original) The program storage medium of claim 25, wherein the encoded method further comprises:

buffering a plurality of written pages; and
writing the plurality of written pages to disk storage in a single write operation.

42. (Original) The program storage medium of claim 25, wherein the encoded method further comprises:

buffering a plurality of written pages; and
writing the plurality of written pages to disk storage in a single write operation.

43. (Previously presented) The program storage medium of claim 25, wherein the encoded method further comprises maintaining cache consistency and wherein maintaining cache consistency comprises:

issuing an exclusive mode token from the server to a client to permit the client to dirty the page;
issuing a shared mode token from the server to the client to permit the client to use, but not dirty, the page;
revoking the exclusive mode token before issuing the shared mode token.

44. (Original) The program storage medium of claim 43, wherein at least one of the exclusive mode token and the shared mode token is embedded in at least one of a read and a write operations in the operating system.

Appl. No. 10/682,041
Amdt. dated May 26, 2005
Reply to Office action of March 17, 2005

45. (Previously presented) The program storage medium of claim 43, wherein revoking the exclusive mode token in the encoded method comprises revoking the exclusive mode token when another computing system wants to read the data.

46. (Original) The program storage medium of claim 43, wherein the encoded method further comprises a server sharing free space information with a plurality of clients.

47. (Canceled).

48. (Previously presented) The program storage medium of claim 25, wherein the encoded method further comprises ensuring that the file modification time a file is updated before or at file close time but not after, comprising:

setting a flag at the client when any page of the file is dirtied;

clearing the flag when dirty data is sent to the server before the file is closed;

notifying the server from the client to update the modification time if the flag is still set at close time; and

notifying the server from the client to forego updating the modification time if dirty data is sent to the server after the file has been closed.

49. (Previously presented) A computer programmed to perform a method for delaying asynchronous writes in a distributed file system to which a unique identifier is assigned, the method comprising:

buffering a page of dirty data with the unique identifier upon writing the data to a server;

changing the unique identifier upon a failure of the server;

comparing the buffered unique identifier with the current unique identifier when the page is requested while the page is in a written state; and

handling the request responsive to the comparison.

Appl. No. 10/682,041
Amdt. dated May 26, 2005
Reply to Office action of March 17, 2005

50. (Original) The programmed computer of claim 49, wherein the method further comprises assigning the unique identifier.

51. (Previously presented) The programmed computer of claim 50, wherein assigning the unique identifier in the programmed method comprises assigning a sequence number.

52. (Original) The programmed computer of claim 51, wherein assigning the sequence number in the programmed method comprises assigning a cluster-wide sequence number.

53. (Original) The programmed computer of claim 51, wherein changing the unique identifier in the programmed method comprises incrementing the sequence number.

54. (Previously presented) The programmed computer of claim 49, wherein buffering the dirty page in the programmed method comprises storing the unique identifier in a header of the page.

55. (Previously presented) The programmed computer of claim 54, wherein buffering the written page with the unique identifier in the programmed method comprises returning the page to a least recently used queue.

56. (Previously presented) The programmed computer of claim 49, wherein buffering the written page with the unique identifier in the programmed method comprises returning the page to a least recently used queue.

57. (Original) The programmed computer of claim 49, wherein the dirty page transitions to a written state upon the write to the server in the programmed method.

**Appl. No. 10/682,041
Amdt. dated May 26, 2005
Reply to Office action of March 17, 2005**

58. (Original) The programmed computer of claim 57, wherein the written page is handled by the programmed method as though in the dirty state when the page is in the written state.

59. (Previously presented) The programmed computer of claim 49, wherein buffering the written page with the unique identifier in the programmed method comprises caching the written page with the unique identifier.

60. (Previously presented) The programmed computer of claim 59, wherein caching the written page with the unique identifier in the programmed method comprises caching the written page with the unique identifier on the client.

61. (Previously presented) The programmed computer of claim 49, wherein changing the unique identifier in the programmed method comprises incrementing the unique identifier.

62. (Previously presented) The programmed computer of claim 49, wherein comparing the buffered unique identifier with the current unique identifier in the programmed method comprises determining whether the current unique identifier is a numerically higher value.

63. (Previously presented) The programmed computer of claim 49, wherein handling the request responsive to the comparison in the programmed method comprises:

storing the buffered written page to disk storage if the buffered unique identifier differs from the current unique identifier; or
performing a file sync operation if the buffered unique identifier matches the current unique identifier.

Appl. No. 10/682,041
Amdt. dated May 26, 2005
Reply to Office action of March 17, 2005

64. (Original) The programmed computer of claim 49, wherein the programmed method further comprises:

buffering a plurality of dirty pages before writing them to the server; and
writing the buffered dirty pages to the server in a single write operation.

65. (Original) The programmed computer of claim 49, wherein the programmed method further comprises:

buffering a plurality of written pages; and
writing the plurality of written pages to disk storage in a single write operation.

66. (Original) The programmed computer of claim 49, wherein the programmed method further comprises:

buffering a plurality of written pages; and
writing the plurality of written pages to disk storage in a single write operation.

67. (Previously presented) The programmed computer of claim 49, wherein the programmed method further comprises maintaining cache consistency and wherein maintaining cache consistency comprises:

issuing an exclusive mode token from the server to a client to permit the client to dirty the page;
issuing a shared mode token from the server to the client to permit the client to use, but not dirty, the page;
revoking the exclusive mode token before issuing the shared mode token.

68. (Original) The programmed computer of claim 67, wherein at least one of the exclusive mode token and the shared mode token is embedded in at least one of a read and a write operations in the operating system.

**Appl. No. 10/682,041
Amdt. dated May 26, 2005
Reply to Office action of March 17, 2005**

69. (Previously presented) The programmed computer of claim 67, wherein revoking the exclusive mode token in the programmed method comprises revoking the exclusive mode token when another computing system wants to read the data.

70. (Original) The programmed computer of claim 67, wherein the programmed method further comprises a server sharing free space information with a plurality of clients.

71. (Canceled).

72. (Previously presented) The programmed computer of claim 49, wherein the programmed method further comprises ensuring that the file modification time a file is updated before or at file close time but not after, comprising:

setting a flag at the client when any page of the file is dirtied;

clearing the flag when dirty data is sent to the server before the file is closed;

notifying the server from the client to update the modification time if the flag is still set at close time; and

notifying the server from the client to forego updating the modification time if dirty data is sent to the server after the file has been closed.